TITLE OF THE INVENTION

System And Method For Providing User Input Information To Multiple Independent, Concurrent Applications

CROSS REFERENCE TO RELATED APPLICATIONS

--None--

STATEMENT REGARDING FEDERALLY SPONSORED RESEARCH OR DEVELOPMENT

--Not Applicable--

BACKGROUND OF THE INVENTION

The present invention is related to the field of user input to computer applications, and in particular to the use of a single input device such as a telephone keypad to provide user input to multiple concurrent computer applications.

Many communications and information services are enabled by telephone access to computer applications. A common mode for interacting with such computer applications is through a touch-tone telephone. Services often connect multiple applications to the user. For example, it is possible for a voice messaging service to be accessed through a pre-paid service. However, this creates a modality problem. For example, the pre-paid platform might wish to know when the user presses and holds the "#" key for a relatively long time (this being referred to as both the "long pound" and the "long octothorpe"), while the voice messaging platform might wish to know when the user enters digits, such as for menu navigation. The modality problem is that all digits entered by the user today get sent to both applications, as the digits are sent and both applications listen to the bearer channel. Each application must be prepared to receive and discard

ATTORNEY DOCKET NO. SNSH-016XX
WEINGARTEN, SCHURGIN,
GAGNEBIN & LEBOVICI LLP
TEL. (617) 542-2290
FAX. (617) 451-0313

Express Mail Number
EV044751269US

notifications of key presses in which it has no interest, complicating the design of the application as well as wasting the use of processing and communications resources during operation.

Numerous applications have been deployed for use in conjunction with the traditional time-division-multiplexed (TDM)-based telephone network. In many cases the applications simply receive the TDM-based "in-band" media stream, i.e., the voice channel, and the applications are responsible for continually decoding the media and monitoring for the presence of certain user input of a signaling nature, such as tones indicating that a particular key on the telephone keypad has been pressed. Certain improvements to the TDM network have been made, such as the Advanced Intelligent Network (AIN), which have the goal of separating signaling traffic from media traffic. However, in practice most of the application logic resides in an "intelligent peripheral" that is coupled along the media path, because of the low-level, device control nature of the associated protocols. There are too many messages with too short a latency budget for a total separation of application logic from the Intelligent Peripheral. The result is that application developers write their applications for deployment on the intelligent peripheral, usually with proprietary intelligent peripheral languages. Thus, AIN does not fulfill the promise of separating application logic from media processing.

The situation is more complex for the packet-switched environments, such as Voice-over-IP. Existing approaches such as H.248.1 (MEGACO), Session Initiation Protocol (SIP) and an in-band technique described in RFC 2833 are described in turn.

H.248.1 (MEGACO) has a provision for reporting key press digits detected or generated by an "endpoint", which in H.248.1 is a media gateway (MG). The MG can be an IP phone, an access gateway, or a trunking gateway. In the case of the IP phone, the IP phone can transmit the key presses directly at the protocol

ATTORNEY DOCKET NO. SNSH-016XX
WEINGARTEN, SCHURGIN,
GAGNEBIN & LEBOVICI LLP
TEL. (617) 542-2290
FAX. (617) 451-0313

level. In the case of a gateway, the gateway can detect the key presses using DTMF detectors. Media Gateway Control Protocol (MGCP) is a proprietary Cisco protocol that operates in much the same manner as H.248.1. These protocols employ a master-slave approach in which a Media Gateway Controller (MGC) commands the MG (using a device control protocol signaling link) to connect a tone detector to an incoming circuit and wait for a digit map match. When the MG detects a key press pattern of interest, it notifies the MGC over the same signaling link, returning the actual digit string detected.

In H.248.1, however, one and only one MGC may control the resources in an MG. Applications that have an interest in user signaling must be a part of the MGC application - there is no provision for independent, third-party applications to receive user signaling information. In MGCP, a first MGC may "pass off" control to a second MGC, but one and only one MGC may control a resource at any given time. The limitation of one and only one controller controlling a resource is a direct result of the master/slave nature of the MGCP and H.248.1 protocols. That is, the protocol requires the MG to be in an exclusive relationship to an MGC. Although these protocols also allow for "virtual MGs" within a physical MG, in which case there may be multiple MGCs serving as masters to the set of virtual MGs in a single physical MG, the virtual MGs are simply partitions of a physical MG. There is no provision for enabling multiple independent applications to selectively obtain user signaling information from a single stream of user input.

It has been proposed that a peer-to-peer protocol such as the Session Initiation Protocol (SIP) be used to transport key press signaling, such as via the SIP INFO method. The proposed mechanism closely follows the protocol of MGCP and H.248.1, including the use of MGCP and H.248.1 messages for specifying digit maps and notifications. However, the proposals have

ATTORNEY DOCKET NO. SNSH-016XX
WEINGARTEN, SCHURGIN,
GAGNEBIN & LEBOVICI LLP
TEL. (617) 542-2290
FAX. (617) 451-0313

envisioned only a single application requesting notifications, which is a result of there being no mechanism for addressing endpoints of interest.

Cisco Systems has introduced a method for transporting DTMF digits using SIP in the SIP signaling path using the SIP NOTIFY method. However, this method has a number of disadvantages. First, notifications can only go to a single egress gateway - it is not possible for a third-party application to register for notifications. Second, the egress gateway receives notifications of every DTMF digit, whether it has an interest in them or not. Third, there is no provision for selectively passing through or clamping the DTMF tones from the media stream. If the ingress gateway passes DTMF, there is the risk of network elements interpreting both the in-band DTMF and the corresponding DTMF signaling received via the NOTIFY mechanism, potentially resulting in incorrect operation.

Another proposed method of transporting key press signaling is to use in-band representations for the keys. For example, RFC 2833 describes transporting key presses as named events, rather than as digital waveforms representing the key presses. While this approach uses less bandwidth and processing resources in the media path, it has serious drawbacks that limit its usability. First, a point-to-point media relationship between the endpoint and the application is generally assumed, leaving no provision for third-party involvement with collecting digits. Although in theory RFC 2833 could be used with third-party applications, rather complicated and unrealistic setup and operation are required. Additionally, because of the particular way that RFC 2833 handles redundancy, it does not meet the reliability requirements for signaling traffic. Moreover, RFC 2833 uses more bandwidth than is necessary, by sending multiple copies of the same packet for normal, lossless operation. Finally, applications receive all key presses, whether they have an interest in the key presses or not,

ATTORNEY DOCKET NO. SNSH-016XX
WEINGARTEN, SCHURGIN,
GAGNEBIN & LEBOVICI LLP
TEL. (617) 542-2290
FAX. (617) 451-0313

making for inefficient use of communication and processing resources.

Thus what is needed is an efficient system and method for detecting and notifying applications of a single mode of user input, such as user key presses, where the user input signaling follows a signaling path distinct from the media path, and that provides for multiple, independent applications to receive the signaling independently.

BRIEF SUMMARY OF THE INVENTION

In accordance with the present invention, a system and method for providing user input information to multiple independent, concurrent applications is disclosed.

The applications generate respective subscription messages that are provided to a device receiving input of a predetermined type from a user, such as key presses on a telephone keypad. The subscription message for each application identifies a respective pattern of the user input that the application is to be notified of. The device may receive the input directly, such as in the case of a telephone that receives key press information directly from the telephone keypad. Alternatively, the device may be of a type, such as a media proxy, that resides along a media path between the user and the application, in which case the device obtains the user input from the media stream.

The device monitors the input from the user to identify the occurrence of the respective patterns identified in the subscription messages. Upon the occurrence of the pattern in a given subscription message, the device notifies the corresponding application via a signaling channel linking the application with the device. In general, the device only notifies the particular application that provided the subscription message, and thus processing and communications resources are conserved.

ATTORNEY DOCKET NO. SNSH-016XX
WEINGARTEN, SCHURGIN,
GAGNEBIN & LEBOVICI LLP
TEL. (617) 542-2290
FAX. (617) 451-0313

The subscription messages use regular expressions that can include various types of formats to specify the patterns of interest. For example, the formats can specify a single digit (either explicitly or in wildcard form), one of a set of digits, a range of digits, and/or a repeating pattern of digits. The subscription messages can also contain tags associated with the regular expressions. When a match is detected and reported to the application by the device, the tag is also returned, enabling the application to easily determine exactly what response to the input is appropriate without needing to maintain a large amount of internal state information.

Other aspects, features, and advantages of the present invention will be apparent from the Detailed Description that follows.

BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWINGS

The invention will be more fully understood by reference to the following Detailed Description of the invention in conjunction with the Drawing, of which:

Figure 1 is a block diagram of a first system for signaling user key press information to applications in accordance with the present invention;

Figure 2 is a block diagram of a second system for signaling user key press information to applications in accordance with the present invention;

Figure 3 is a block diagram of a third system for signaling user key press information to applications in accordance with the present invention;

Figure 4 is a block diagram of a fourth system for signaling user key press information to applications in accordance with the present invention;

Figure 5 is a diagram of a key press buffer and associated pointers used in any of the systems shown in Figure 1-4; and

ATTORNEY DOCKET NO. SNSH-016XX
WEINGARTEN, SCHURGIN,
GAGNEBIN & LEBOVICI LLP
TEL. (617) 542-2290
FAX. (617) 451-0313

Figure 6 is a diagram of a schema for a key press markup language (KPML) used in any of the systems shown in Figure 1-4.

DETAILED DESCRIPTION OF THE INVENTION

Figure 1 shows a first embodiment in which an application 10 (which is a SIP User Agent (UA)) directly interacts, on a given two-party dialog, with an "end point device" 12. The application 10 is connected to the end point device 12 by both a SIP connection 14 (for signaling) and one or more Real-Time Protocol (RTP) connections 16 (for media). In this scenario, the application 10 requests that the end point device 12 report via the SIP connection 14 on key press events that might normally emanate from the end point device's RTP port 18. The illustrated configuration can represent, for example, a toll by-pass scenario where the end point device 12 is an ingress gateway and the application 10 is an egress gateway.

In the case shown in Figure 1, the application 10 requests digit notification on the same dialog established for the call, between SIP ports 20 and 22. The use of SIP and RTP in Figure 1 is for exemplary purposes only. Other signaling mechanisms such as H.323 can be used, and other bearer mechanisms such as ATM or TDM can be used.

In general, the end point device 12 may receive the input directly, such as in the case of a telephone that receives key press information directly from the telephone keypad. Alternatively, the device may be of a type, such as a media proxy, that resides along a media path between the user and the application, in which case the end point device 12 obtains the user input from the media stream. An example of such a configuration is shown below. In general, the end point device 12 includes hardware and software processing resources and interfaces in accordance with its function in the system. In particular, the end point device 12 includes hardware for monitoring the user

ATTORNEY DOCKET NO. SNSH-016XX
WEINGARTEN, SCHURGIN,
GAGNEBIN & LEBOVICI LLP
TEL. (617) 542-2290
FAX. (617) 451-0313

input to detect patterns of interest, as described in more detail below, and one or more processors programmed to implement the signaling functionality described herein.

Figure 2 shows a second embodiment that supports a third-party application 24 that is interested in user key presses occurring in the context of an established two-party SIP dialog between the end point device 12 and the first application 10. The third party application 24 addresses the particular media stream by referencing an established SIP dialog identifier that refers to the dialog between the SIP ports 20 and 22.

Figure 3 shows a third embodiment employing a "media proxy" 26 that monitors the media stream among a plurality of endpoints. A media proxy is a device that forwards media streams under the control of a signaling component that provides, at a minimum, instruction as to the source and destination addresses. In addition to the media forwarding function, the media proxy 26 can also do light media processing, such as tone detection. An example of a media proxy is the A1-G2 MF from SnowShore Networks, Inc.

In Figure 3, the media proxy 26 has established connections to a first endpoint User Agent UAa 28 and a second endpoint User Agent UAb 30. A requesting application 32 uses dialog identifiers to identify the stream to monitor. The default is to monitor the media entering the end point device. For example, if the requesting application 32 uses the dialog represented by SIP ports 34 and 36, then the media coming from UAa RTP port 38 is monitored. Likewise, specifying the dialog represented by ports 40 and 42 directs the media proxy 26 to monitor the media coming from UAb RTP Port 44. The requesting application 32 can monitor the reverse direction or other, related streams. There could also be multiple streams if there are multiple audio sources, for example.

ATTORNEY DOCKET NO. SNSH-016XX
WEINGARTEN, SCHURGIN,
GAGNEBIN & LEBOVICI LLP
TEL. (617) 542-2290
FAX. (617) 451-0313

As shown in Figure 4, a plurality of applications may be interested in receiving different notifications of digit map patterns. Applications 46 and 48 make requests directly to, and receive reports directly from, an end point device 50. Other applications 52-1, 52-2, ...52-n register with a controlling, or "aggregation", server 54, which collapses the various requests into as few as one KPML request to the end point device 50.

One problem with employing an aggregation server 54 is its potential to become a bottleneck, because it must process all of the requests and forward them on to the end point device 50. Another problem is that some patterns may have complex inter-key-press timing relationships within a pattern, but not across independent pattern requests, thus making it difficult to specify values for the various timers. Finally, the aggregation server 54 must disambiguate and forward responses appropriately in the face of conflicting or overlapping expressions from the different requesting applications 52. Thus, to avoid such problems, it is generally preferred that the individual requesting applications, such as applications 46 and 48 in Figure 4, make requests directly to an end point device such as end point device 50. Note that this approach still supports the aggregation model presented by the aggregation server 54.

In addition to reducing network traffic, one of the goals of the presently disclosed technique is to reduce processing requirements at the requesting applications. For example, assume the first requesting application 46 of Figure 4 is looking for "***", while the second requesting application 48 is looking for "L#", the octothorpe key pressed for a time. When the end point device 50 detects the "***" pattern, it would be inefficient to send a report to the second requesting application 48. Thus, the end point device 50 sends the notification only to the first requesting application 46.

ATTORNEY DOCKET NO. SNSH-016XX
WEINGARTEN, SCHURGIN,
GAGNEBIN & LEBOVICI LLP
TEL. (617) 542-2290
FAX. (617) 451-0313

The system described herein uses a subscription protocol mechanism, whereby a party such as the application server 32 subscribes to the key press state of a device such as the media proxy 26 or end point device 12. The SIP SUBSCRIBE/NOTIFY mechanism, as described in RFC 3265, can be employed. If media sessions are set up using the SIP INVITE mechanism, as described in RFC 3261, then requesting applications can use the SIP call identifiers to identify the media leg to be monitored. In alternative embodiments, session description protocol (SDP) identifiers such as Internet Protocol address and Port numbers can be used.

The SIP SUBSCRIBE/NOTIFY mechanism provides a means for handling multiple, independent requests. Namely, subscriptions on different SIP dialogs are independent. In addition, a subscription on a particular dialog, with a unique event identifier tag (the "id" tag to the "event" entity in the SUBSCRIBE request) is also an independent subscription for the purposes of key press handling and reporting.

SUBSCRIBE/NOTIFY protocols provide subscription state management. Following the mechanics of RFC 3261, if an end point device 12 receives a SUBSCRIBE request on an existing subscription, the end point device 12 unloads the current subscription and replaces it with the new one.

One goal of the presently disclosed system is to reduce network traffic by consolidating multiple digit presses into one notification message. The system can take advantage of the fact that many telephony applications are interested in not only a single key press, but multiple key presses. For example, collecting a North American Numbering Plan telephone number requires collecting 10 digits, while collecting a Personal Identification Number (PIN) code may require collecting 4 to 6 digits. The system achieves this goal by having the requesting application include regular expressions describing the patterns to

ATTORNEY DOCKET NO. SNSH-016XX
WEINGARTEN, SCHURGIN,
GAGNEBIN & LEBOVICI LLP
TEL. (617) 542-2290
FAX. (617) 451-0313

collect. The following section describes one embodiment of a regular expression syntax that can be employed. This syntax is referred to as DRegex, for digit regular expression. It should be noted that "white space" is removed before DRegex is parsed, which enables sensible printing in XML without affecting the meaning of the DRegex string.

Table 1 describes the use of DRegex. Table 2 gives some examples of DRegex regular expression formats.

Format	Matches
digit	single digit 0-9 and A-D
#	# key
*	* key
[digit selector]	Any digit in selector
[^digit selector]	Any digit NOT in selector
[digit-range]	Any digit in range
x	Any digit 0-9
.	Zero or more repetitions of previous pattern
|	Alternation
{m}	m repetitions of previous pattern
{m,}	m or more repetitions of previous pattern
{,n}	At most n (including zero) repetitions of previous pattern
{m,n}	at least m and at most n repetitions of previous pattern
Lc	Match the character c if it is "long"; c is a digit, #, or *.

Table 1 - DRegex Formats

ATTORNEY DOCKET NO. SNSH-016XX
WEINGARTEN, SCHURGIN,
GAGNEBIN & LEBOVICI LLP
TEL. (617) 542-2290
FAX. (617) 451-0313

Example	Description
1	Matches the digit 1
[179]	Matches 1, 7, or 9
[^01]	Matches 2, 3, 4, 5, 6, 7, 8, 9
[2-9]	Matches 2, 3, 4, 5, 6, 7, 8, 9
x	Matches 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
2|3	Matches 2 or 3; same as [23]
00|011	Matches the string 00 or 011
0.	Zero or more occurrences of 0
[2-9].	Zero or more occurrences of 2-9
*6[179#]	Matches *61, *67, *69, or *6#
011x{7,15}	011 followed by seven to fifteen digits
L*	Long star

Table 2 - DRegex Examples

Referring again to Figure 4, a first requesting application 46 requests that an end point device 50 report detected key presses. The requesting application 46 registers for a pattern, such as "***" (three stars in succession), using the SUBSCRIBE mechanism. When the end point device 50 detects the pattern, it sends a NOTIFY message to the requesting application 46, noting the detection of the pattern. Example 1 shows a request, using an embodiment of a protocol language referred to as the Keypress Markup Language (KPML).

ATTORNEY DOCKET NO. SNSH-016XX
WEINGARTEN, SCHURGIN,
GAGNEBIN & LEBOVICI LLP
TEL. (617) 542-2290
FAX. (617) 451-0313

```
<?xml version="1.0" encoding="UTF-8"?>
<kpml xmlns="urn:ietf:params:xml:ns:kpml"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="urn:ietf:params:xml:ns:kpml kpml.xsd"
    version="1.0">
  <request>
    <pattern>
      <regex>***</regex>
    </pattern>
  </request>
</kpml>
```

Example 1 - Single Regular Expression Request

The message in Example 1 is a request registering a set of patterns, of which there is only a single regular expression, that of the three stars ("***").

A requesting application can also register for multiple patterns. For example, the requesting application could look for any of a set of patterns "***", "*1", "*61", and "*62", as shown in Example 2.

ATTORNEY DOCKET NO. SNSH-016XX
WEINGARTEN, SCHURGIN,
GAGNEBIN & LEBOVICI LLP
TEL. (617) 542-2290
FAX. (617) 451-0313

```
<?xml version="1.0" encoding="UTF-8"?>
<kpml xmlns="urn:ietf:params:xml:ns:kpml"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="urn:ietf:params:xml:ns:kpml kpml.xsd"
      version="1.0">
   <request>
      <pattern>
         <regex>***</regex>
         <regex>*1</regex>
         <regex>*61</regex>
         <regex>*62</regex>
      </pattern>
   </request>
</kpml>
```

Example 2 - Multiple Regular Expressions Request

There are different techniques that can be used to determine what pattern matches a given key press string, including for example longest match, shortest match, or most specific match. The matching algorithm to use can be specified in the KPML message. Although in many cases the longest match will be preferable, this technique has the problem that the system will always wait for the next digit, even if the best match has occurred. That is, all key press collection events end with a timeout.

To improve the user experience by having fast reporting of a match, while maintaining the longest match property, the system uses a set of special timers and the specification of an Enter Key Sequence. The timers are the critical timer, the inter-digit timer, and the extra digit timer. The critical timer is the time to wait for another digit if the collected digits can match a pattern. The extra timer is the time to wait after the longest match has occurred (presumably for the return key). The inter-

ATTORNEY DOCKET NO. SNSH-016XX
WEINGARTEN, SCHURGIN,
GAGNEBIN & LEBOVICI LLP
TEL. (617) 542-2290
FAX. (617) 451-0313

digit timer is the time to wait between digits in all other cases. Note there is no start timer, as that concept does not apply in the KPML context.

The Enter Key Sequence is a method whereby one specifies a key press or string of key presses that indicates to the system that entry is complete. The system reports immediately with a match or no match error upon receipt of the Enter Key Sequence. For convenience, the system does not send the Enter Key Sequence in the report. This saves the requesting application from having to trim irrelevant information. Note that the Enter Key Sequence cannot also be a substring in the regular expression.

For patterns such as the long octothorpe, applications instruct the device what constitutes "long" by setting a "longtimer" attribute in the <pattern> tag to the number of milliseconds desired. Some phones, particularly private branch exchange (PBX) phones, transmit a digit for a preset length of time, such as 50ms, irrespective of how long the user presses the respective key. To enable applications to respond to said devices, applications may indicate to the device to consider a repetition of a key press within the longtime time period. Applications indicate this behavior to the device by setting the longrepeat attribute to "true" in the <pattern> tag.

Example 3 shows an example of a request that registers for a long octothorpe. With a longtimer value of 2000, the device will look for a minimum key press duration of two seconds (2000 milliseconds). This example also directs the device to match a succession of octothorpe keys for two seconds as long key press duration by setting longrepeat to true.

ATTORNEY DOCKET NO. SNSH-016XX
WEINGARTEN, SCHURGIN,
GAGNEBIN & LEBOVICI LLP
TEL. (617) 542-2290
FAX. (617) 451-0313

```
<?xml version="1.0" encoding="UTF-8"?>
<kpml xmlns="urn:ietf:params:xml:ns:kpml"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:ietf:params:xml:ns:kpml kpml.xsd"
 version="1.0">
  <request>
    <pattern longtimer="2000" longrepeat="true">
      <regex>L#</regex>
    </pattern>
  </request>
</kpml>
```

Example 3 - Long Octothorpe Request

Some applications care to continuously monitor the stream for a particular pattern, while other applications look for only a single occurrence of a particular pattern, at which time the application is finished monitoring the end point device or may register a different set of patterns. The first type of request is referred to as a "persistent" request, while the second type is termed a "one-shot" request. The system provides for the requesting application to specify the nature (persistent or one-shot) of the request.

It is advantageous to refrain from notifying applications of key presses that are not of interest. However, it is possible that a human user may press spurious keys or accidentally press an incorrect key. Applications need to perform error recovery in this situation. The disclosed system addresses this problem by starting an inter-digit timer upon detection of the first key that matches the first character of any regular expression in a subscription. This inter-digit timer restarts after every key press detected. If the inter-digit timer expires, the end point device sends a failure notification to the requesting application(s) for which the key presses started to match patterns. This notification

ATTORNEY DOCKET NO. SNSH-016XX
WEINGARTEN, SCHURGIN,
GAGNEBIN & LEBOVICI LLP
TEL. (617) 542-2290
FAX. (617) 451-0313

enables the requesting application to take different actions based on the incomplete key presses.

Another situation that can occur is for the end point device to start collecting key press events and the media dialog to terminate before a pattern matches. Again, the end point device sends the key presses collected up to the point of the dialog termination to the requesting application.

The system and protocol described herein could use any of a variety of message transport mechanisms. As described herein, one preferred transport employs SIP SUBSCRIBE and NOTIFY requests to transport an XML markup called KPML. Those skilled in the art could derive alternate representations for the protocol messages described herein, such as ASN.1 notation for example.

As shown in Example 3, a KPML message contains a <request> entity that includes a <pattern> tag with a series of <regex> elements. The <regex> element specifies a digit pattern for the device to report on. Because a requesting application may need to know which <regex> of a plurality of <regex>s matched, KPML supports a tag attribute to the <regex> element. When there is a match, the report from the end point device includes the tag of the match's <regex> element. Example 4 shows the case where a seven-digit number matches the "local" tag, a 1+ten-digit number matches the "ld" (or long distance) tag, and a 011 followed by a number and five to fifteen digits matches the "iddd" (or international direct distance dialing) tag. A sample response is in Example 5.

ATTORNEY DOCKET NO. SNSH-016XX
WEINGARTEN, SCHURGIN,
GAGNEBIN & LEBOVICI LLP
TEL. (617) 542-2290
FAX. (617) 451-0313

```
<?xml version="1.0" encoding="UTF-8"?>
<kpml xmlns="urn:ietf:params:xml:ns:kpml"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:ietf:params:xml:ns:kpml kpml.xsd"
 version="1.0">
  <request>
    <pattern extradigittimer="1000" interdigittimer="5000">
      <regex tag="local">x{7}</regex>
      <regex tag="ld">1x{10}</regex>
      <regex tag="iddd">011x{5,15}</regex>
    </pattern>
  </request>
</kpml>
```

Example 4 - Use of a Tag to Identify Which Pattern Matched

```
<?xml version="1.0" encoding="UTF-8"?>
<kpml xmlns="urn:ietf:params:xml:ns:kpml"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:ietf:params:xml:ns:kpml kpml.xsd"
 version="1.0">
  <response code="200" text="OK" digits="17035551212" tag="ld"/>
</kpml>
```

Example 5 - Tagged Response

A requesting application may advantageously use the tag attribute to encode, for example, state information. In this way, the requesting application can be a stateless, stimulus driven application. By using the tag attribute to carry state, the requesting application need not carry per-session state. This results in a significant savings in memory and processing requirements at the requesting application, thus offering higher scale and performance for the application.

ATTORNEY DOCKET NO. SNSH-016XX
WEINGARTEN, SCHURGIN,
GAGNEBIN & LEBOVICI LLP
TEL. (617) 542-2290
FAX. (617) 451-0313

For example, Example 6 shows the request for a first device, while Example 7 shows the request for a second device. The server can tell the different requests apart from the unique prefix, in this case "id00" and "id01" for Example 6 and Example 7, respectively. Note that the tags are opaque strings and can contain however much state information the client requires. In the examples here, the identifiers "id00" and "id01" are enough to associate the results with a session. The identifiers can have any arbitrary state information that is meaningful to the application. For example, a tag might be "from-host-192.168.1.12 at step 24 cookie='23d0ij32d0ioicq3icoiqwjemf'", representing host information, sequence information, and arbitrary data.

```
<?xml version="1.0" encoding="UTF-8"?>
<kpml xmlns="urn:ietf:params:xml:ns:kpml"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:ietf:params:xml:ns:kpml kpml.xsd"
 version="1.0">
  <request>
    <pattern extradigittimer="1000" interdigittimer="5000">
      <regex tag="id00-local">x{7}</regex>
      <regex tag="id00-ld">1x{10}</regex>
      <regex tag="id00-iddd">011x{5,15}</regex>
    </pattern>
  </request>
</kpml>
```

Example 6 - Use of a Tag to Identify Server State (Part 1)

ATTORNEY DOCKET NO. SNSH-016XX
WEINGARTEN, SCHURGIN,
GAGNEBIN & LEBOVICI LLP
TEL. (617) 542-2290
FAX. (617) 451-0313

```
<?xml version="1.0" encoding="UTF-8"?>
<kpml xmlns="urn:ietf:params:xml:ns:kpml"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:ietf:params:xml:ns:kpml kpml.xsd"
 version="1.0">
  <request>
    <pattern extradigittimer="1000" interdigittimer="5000">
      <regex tag="id01-local">x{7}</regex>
      <regex tag="id01-ld">1x{10}</regex>
      <regex tag="id01-iddd">011x{5,15}</regex>
    </pattern>
  </request>
</kpml>
```

Example 7 - Use of a Tag to Identify Server State (Part 2)

Some devices buffer entered digits, and applications behave differently with respect to such buffered digits. Some applications may require that a user enter key presses only as the application is ready to receive them, in which case any key presses entered prematurely should be ignored or discarded. Other applications enable a user to "barge" through prompts or other delays in the user interface by entering key presses in advance of their prompts, for example, and then acting on the buffered key presses rather than discarding them.

In the disclosed system, the default is that the matching specified by new requests is carried out with respect to buffered digits first, such operation enabling "barging" behavior by the user. However, the protocol provides a <flush> tag in the <pattern> element to specify whether the buffer should be flushed before any matching operations begin, which has the effect of ignoring any digits entered before receipt of the KPML request. Additional detail on the digit buffering mechanism is provided

ATTORNEY DOCKET NO. SNSH-016XX
WEINGARTEN, SCHURGIN,
GAGNEBIN & LEBOVICI LLP
TEL. (617) 542-2290
FAX. (617) 451-0313

below in conjunction with a description of key press "quarantining."

As mentioned above, an end point device may support an inter-digit timeout value, which is the amount of time the end point device waits for user input before returning a timeout error result on a partially matched pattern. The application can specify the inter-digit timeout as an integer number of milliseconds by using an inter-digit timer attribute to the <pattern> tag. The default is 4000 milliseconds. If the end point device does not support the specification of an inter-digit timeout, the end point device silently ignores the request. If the end point device supports the specification of an inter-digit timeout, but not to the granularity specified by the value presented, the end point device rounds up the requested value to the closest value it can support.

KPML messages are independent, a property that enables multiple requesting applications to simultaneously monitor a stream without interacting with each other. One result of this independence, however, is that it is not possible for a requesting application to know whether a following request from another application will enable barging or want the digits flushed. Therefore, the end point device quarantines all digits detected between the time of a notification and the interpretation of the next request, if any. If the next request indicates a buffer flush, then the end point device flushes all collected digits from consideration from KPML requests received on that dialog with the given event id. If the next request does not indicate that buffered digits should be flushed, then the end point device applies the buffered digits (if possible) against the digit maps presented by the request's <regex> tags. If there is a match, the end point device issues the appropriate notification. If there is

ATTORNEY DOCKET NO. SNSH-016XX
WEINGARTEN, SCHURGIN,
GAGNEBIN & LEBOVICI LLP
TEL. (617) 542-2290
FAX. (617) 451-0313

no match, the end point device flushes all of the collected digits on that request's buffer.

By default, the end point device transmits in-band tones (RFC 2833 events or actual tones) on the media channel in parallel with digit reporting via the signaling channel. Note that in the absence of this behavior, a user device could easily break called applications. For example, consider a personal assistant application that uses "*9" for attention. If the user presses the "*" key, the device holds the digit looking for the "9". However, the user may enter another "*" key, possibly because they accessed an IVR system that looks for "*". In this case, the "*" would get held by the device, because it is looking for the "*9" pattern. The user would probably press the "*" key again, hoping that the called IVR system just did not hear the key press. At that point, the user device would send both "*" entries, as "**" does not match "*9". However, that would not have the effect the user intended when they pressed "*".

On the other hand, there are situations where passing through tones in-band is not desirable. Such situations include call centers that use in-band tone spills to effect a transfer. For those situations, a digit suppression tag "pre" can be used in conjunction with the <regex> tag to indicate that the transmission of digits in the media stream should be suppressed. There can only be one <pre> in any given <regex>. An example of a request including a suppression tag is shown in Example 8.

If there is only a single <pattern> and a single <regex>, suppression processing is straightforward. The end point device passes digits until the stream matches the regular expression <pre>. At that point, the end point device will continue collecting digits, but will suppress the generation or pass-through of any in-band digits. When reporting on a match, the end point device will indicate whether it suppressed any digits by

ATTORNEY DOCKET NO. SNSH-016XX
WEINGARTEN, SCHURGIN,
GAGNEBIN & LEBOVICI LLP
TEL. (617) 542-2290
FAX. (617) 451-0313

including an attribute "suppressed" with a value of "true" in the digit report.

```
<?xml version="1.0" encoding="UTF-8"?>
<kpml xmlns="urn:ietf:params:xml:ns:kpml"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:ietf:params:xml:ns:kpml kpml.xsd"
 version="1.0">
  <request>
    <pattern>
      <enterkey>#</enterkey>
      <regex><pre>*8</pre>x{7}|x{10}</regex>
      <regex>*6</regex>
    </pattern>
  </request>
</kpml>
```

Example 8 - Digit Suppression

In Example 8, the end point device will begin to suppress digits after receiving the string '*8'. It then looks for seven or ten digits. If it receives something other than a digit, receives the enter key sequence, or times out, then the end point device will return an error report. Note the second pattern, <regex>*6</regex> does not start the digit suppression method.

Digit suppression can be optional for a device. If a device is not capable of digit suppression, it should ignore the digit suppression attribute and never send a suppressed indication in the digit report. In this case, it will match concatenated patterns of pre+value.

At some point during digit suppression, the end point device will collect enough digits to the point it hits a <pre> pattern. The inter-digit timer attribute indicates how long to wait once the user enters digits before reporting a time-out error. If the

ATTORNEY DOCKET NO. SNSH-016XX
WEINGARTEN, SCHURGIN,
GAGNEBIN & LEBOVICI LLP
TEL. (617) 542-2290
FAX. (617) 451-0313

inter-digit timer expires, the endpoint issues a time-out report and transmits the suppressed digits on the media stream.

Once the end point device detects a match and it sends a NOTIFY request to report the digit string, the end point device stops digit suppression. Clearly, if subsequent digits match another <pre> expression, then the end device starts digit suppression again.

After digit suppression begins, it might become clear that a match will not occur. For example, take the expression "<regex><pre>*8</pre>xxx[2-9]xxxxxx</regex>". At the point the endpoint receives "*8", it will stop sending digits. Let us say that the next three digits from the user are "408", which match the "xxx" part of the pattern. If the next digit is a zero or one, the pattern will not match, meaning that there is no mechanism for re-enabling the sending of digits on the media connection.

Applications should be very careful to indicate suppression only when they are fairly sure the user will enter a digit string that will match the regular expression. In addition, applications should deal with situations such as no-match or time-out. This is because the endpoint will hold digits, which will have obvious user interface issues in the case of a failure. Also, it is very important for the endpoint to have a sensible inter-digit timer. This is because an errant dot (".") may suppress digit sending forever. Reasonable values are on the order of four seconds.

One unique aspect of the presently disclosed system is the handling of quarantined key presses. A quarantined key press is an event that does not yet match a regular expression. The system often quarantines key presses between the time of a notification and, particularly in the case of one-shot subscriptions, the receipt of a new subscription.

If the user presses a key not matched by the <regex> tags, the end point device discards the key press from consideration against the current or future KPML messages. However, once there

ATTORNEY DOCKET NO. SNSH-016XX
WEINGARTEN, SCHURGIN,
GAGNEBIN & LEBOVICI LLP
TEL. (617) 542-2290
FAX. (617) 451-0313

is a match, the end point device quarantines any key presses the user entered subsequent to the match. This behavior allows for applications to only receive digits that interest them.

Before the receipt of the first subscription, the end point device does not normally quarantine key press events. This can create a race condition. Namely, the end point device can establish a session and have key press events occur before the end point device receives the first subscription. If this is of concern, the end point device can be provisioned to quarantine all key press events, or the end point device can be signaled at session establishment time that there will be a subscription. In the SIP environment, the first KPML subscription can be included as a separate MIME body part to the initial INVITE message.

At some point in time, the end point device has an indication to quarantine digits, either because of provisioning, explicit signaling, or receiving a subscription. That subscription will be associated with a SIP dialog or dialog/id pair. In the case of explicit provisioning or explicit signaling, there will be an implicit subscription on the INVITE dialog. In the case of a SUBSCRIBE request, the subscription is associated with the given SUBSCIBE-initiated dialog. Again, the end point device considers a SUBSCRIBE request with a unique "id" tag to be a new subscription for quarantine purposes.

The importance of separate subscriptions comes from the per-subscription quarantine of key presses. This enables application consistency. The application will get all key presses, in temporal order, which it is interested in, without interference from the subscriptions of other applications.

It could be required that the current request determine the quarantine policy for post-notification processing. This makes sense when all subscriptions come from a single application, which presumably is aware of its own quarantine needs. However, such an

ATTORNEY DOCKET NO. SNSH-016XX
WEINGARTEN, SCHURGIN,
GAGNEBIN & LEBOVICI LLP
TEL. (617) 542-2290
FAX. (617) 451-0313

approach is not acceptable for a system that must support multiple independent applications, each potentially unaware of the others. Each request must set its own quarantine policy for the key presses already buffered.

Requests can explicitly flush the per-subscription key press buffer when the end point device loads a new KPML request on the given subscription by specifying the <flush>yes</flush> entity in the <pattern> tag. Also, requests can explicitly flush digits in the current subscription's buffer that have already been matched or considered by other subscriptions by specifying the <flush>others</flush> entity in the <pattern> tag.

Figure 5 shows a single key press buffer 56 for recording all key presses, with per-stream indices or pointers 58, 60 and 62 into the key press buffer indicating where in the buffer the stream is. Examples are given to illustrate operation of this buffer 56.

Consider the pattern <regex>xxxxxxxx</regex> specified by subscription 1. In this case, the buffered digits 12345678 match the pattern. Thus, the pattern succeeds, the end point device reports the match, and the pointer 58 is subsequently positioned at buffer location 64.

Next, consider the pattern <regex>x{10}</regex>. In this case, the collected digits 123456789# will never match the pattern, because "#" is not a digit and will not match 'x'. The pattern fails, the end point device reports the failure, and the pointer 58 is subsequently positioned at buffer location 66.

Finally, consider the pattern <regex>***</regex>. In this case, the end point device scans the buffer until the first * at position 66, which is the first character that matches the pattern, and thus bypasses the first 10 key presses entered. The pattern then succeeds, the end point device reports the match, and the pointer 58 will be positioned at buffer location 68.

ATTORNEY DOCKET NO. SNSH-016XX
WEINGARTEN, SCHURGIN,
GAGNEBIN & LEBOVICI LLP
TEL. (617) 542-2290
FAX. (617) 451-0313

Now consider what happens if a request specifies <flush>others</flush> and a <regex>x{7}</regex>. In this case, other streams have seen key presses up to position 68, the position of buffer pointer 62. Thus the end point device moves the buffer pointer 58 to position 68 and then starts matching key presses. In this example, the pattern fails, as there are only three digits before a non-digit key (the long #). Because the end point device has considered all key presses between the positions pointed to by pointers 58 and 60, the end point device frees the memory in the buffer between those positions.

The key press buffer can be made quite large, for example by employing a large secondary storage device in the form of a hard disk. The end point device can put parts of the key press buffer onto the secondary storage device, keeping more recently referenced buffered key presses in main memory. This operation is accomplished in a straightforward manner using conventional virtual memory techniques.

Even with a large secondary storage device, it is possible to fill the buffer completely, especially if a large number of media streams are being monitored and if many streams have multiple subscriptions. To address this problem, the end point device can use a circular buffer. When the end-of-buffer pointer will overlap a given subscription's pointer, the subscription's pointer is set to the next subscription pointer, in increasing time order. The end point device also sets a "forced_flush" indicator for the subscription pointer, which is included in the end point device's report on a match or failure and then subsequently cleared.

Note that using a circular buffer is superior to having key presses age out of the buffer. This is because there is no reason to remove key press history if the memory is available. Likewise, under load, the buffer may still overflow if requests do not free

ATTORNEY DOCKET NO. SNSH-016XX
WEINGARTEN, SCHURGIN,
GAGNEBIN & LEBOVICI LLP
TEL. (617) 542-2290
FAX. (617) 451-0313

digits off of the buffer fast enough. Nonetheless, the presently disclosed technique may be used with non-circular buffers as well.

When the user enters key press(es) that match a <regex> tag, the end point device will issue a report. After reporting, the end point device terminates the KPML session unless the subscription has a persistence indicator. If the subscription does not have a persistence indicator, the end point device sets the state of the subscription to "terminated" in the NOTIFY report. If the requestor desires to collect more digits, it must issue a new request. If the subscription has a persistence indicator, then the device uses the same <pattern> to match against future key presses.

KPML reports have two mandatory attributes, code and text. These attributes describe the state of a KPML interpreter on the end device. In the preferred embodiment, the SIP state of the subscription, such as "active" or "terminated", is also a protocol parameter. If one were to use a transport other than SIP that does not convey the subscription state, the subscription state must be one of the attributes of the KPML report.

Note that the KPML code is not necessarily related to the SIP result code. An important example is when a legal SIP subscription request gets a normal SIP 200 OK followed by a NOTIFY, but there is something wrong with the KPML request. In this case, the NOTIFY would include the KPML failure code in the KPML report. Note that from a SIP perspective, the SUBSCRIBE and NOTIFY were successful. Also, if the KPML failure is not recoverable, the end device will most likely set the Subscription-State to terminated. This lets the SIP machinery know the subscription is no longer active.

If a pattern matches, the end point device emits a KPML report. Since this is a success report, the code is "200" and the text is "OK". The KPML report includes the actual digits matched in the digit attribute. The digit string uses the conventional

ATTORNEY DOCKET NO. SNSH-016XX
WEINGARTEN, SCHURGIN,
GAGNEBIN & LEBOVICI LLP
TEL. (617) 542-2290
FAX. (617) 451-0313

characters '*' and '#' for star and octothorpe, respectively. The KPML report also includes the tag attribute if the regex that matched the digits had a tag attribute. If the subscription requested digit suppression and the end device suppressed digits, the suppressed attribute indicates "true". The default value of suppressed is "false". Embodiments of the device may have a datum indicating if the device terminated digit collection due to the receipt of the Enter Key sequence in the KPML report. It may be desirable to omit this datum, as many applications do not need this information.

Example 9 shows a typical KPML response. This is an example of a response to the request shown in Example 4.

```
<?xml version="1.0" encoding="UTF-8"?>
<kpml xmlns="urn:ietf:params:xml:ns:kpml"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:ietf:params:xml:ns:kpml kpml.xsd"
 version="1.0">
  <response code="200" text="OK" tag="1d" digits="17035551212"/>
</kpml>
```

Example 9 - Sample Response

There are a few circumstances in which the end point device will emit a no match report. They are an immediate NOTIFY in response to SUBSCRIBE request (no digits detected yet), a request for service not supported by the end device, or a failure of a digit map to match a string (timeout). Preferably, the NOTIFY in response to a SUBSCRIBE request has no KPML if there are no matching quarantined digits.

Example 10 shows a typical time-out response. Note the device reports the digits collected up to the timeout in the digits attribute of the response tag.

ATTORNEY DOCKET NO. SNSH-016XX
WEINGARTEN, SCHURGIN,
GAGNEBIN & LEBOVICI LLP
TEL. (617) 542-2290
FAX. (617) 451-0313

```
<?xml version="1.0" encoding="UTF-8"?>
<kpml xmlns="urn:ietf:params:xml:ns:kpml"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:ietf:params:xml:ns:kpml kpml.xsd"
 version="1.0">
  <response code="423" text="Timer Expired" digits="1703555"/>
</kpml>
```

Example 10 - Sample Time-Out Response

If there are quarantined digits in the SUBSCRIBE request that match a pattern, then the NOTIFY message in response to the SUBSCRIBE request includes the appropriate KPML document.

Key presses collected using the disclosed method can contain sensitive information, such as PIN codes, credit card numbers, or other personal information. Thus notifications should be secure, integrity checked, and not accessible to unauthorized parties. Likewise, the end point device needs to authenticate subscriptions and make sure the subscriber is authorized to access the key press information. Moreover, subscriptions also need cryptographic integrity guarantees and need protection against spoofing and man-in-the-middle attacks.

A technology known as Transport Layer Security (TLS) can be used for transport security, and a separate technology known as S/MIME can be used for the SUBSCRIBE and NOTIFY methods to guarantee authentication, integrity, and non-repudiation.

Example 10 is an example of a schema that embodies the facets of the preferred embodiment as described herein. A graphical representation of this schema is in Figure 6.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="urn:ietf:params:xml:ns:kpml"
 xmlns="urn:ietf:params:xml:ns:kpml"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
```


ATTORNEY DOCKET NO. SNSH-016XX
WEINGARTEN, SCHURGIN,
GAGNEBIN & LEBOVICI LLP
TEL. (617) 542-2290
FAX. (617) 451-0313

```
            elementFormDefault="qualified"
            attributeFormDefault="unqualified">
        <xs:element name="kpml">
         <xs:annotation>
          <xs:documentation>IETF Keypad Markup
                                Language</xs:documentation>
         </xs:annotation>
         <xs:complexType>
          <xs:choice>
           <xs:element name="request">
            <xs:complexType>
             <xs:sequence>
              <xs:element name="stream" type="xs:string"
                                  minOccurs="0"/>
              <xs:element name="pattern">
               <xs:complexType>
                <xs:sequence>
                 <xs:element name="flush" minOccurs="0">
                  <xs:complexType>
                   <xs:simpleContent>
                    <xs:extension base="xs:string"/>
                   </xs:simpleContent>
                  </xs:complexType>
                 </xs:element>
                 <xs:element name="enterkey" minOccurs="0">
                  <xs:complexType>
                   <xs:simpleContent>
                    <xs:extension base="xs:string"/>
                   </xs:simpleContent>
                  </xs:complexType>
                 </xs:element>
                 <xs:element name="regex" maxOccurs="unbounded">
                  <xs:complexType mixed="true">
```


ATTORNEY DOCKET NO. SNSH-016XX
WEINGARTEN, SCHURGIN,
GAGNEBIN & LEBOVICI LLP
TEL. (617) 542-2290
FAX. (617) 451-0313

```
                    <xs:sequence>
                     <xs:element name="pre" minOccurs="0">
                      <xs:complexType>
                       <xs:simpleContent>
                        <xs:extension base="xs:string"/>
                       </xs:simpleContent>
                      </xs:complexType>
                     </xs:element>
                    </xs:sequence>
                    <xs:attribute name="tag" type="xs:string"
                                             use="optional"/>
                   </xs:complexType>
                  </xs:element>
                 </xs:sequence>
                 <xs:attribute name="persistent" type="xs:boolean"
                               use="optional"/>
                 <xs:attribute name="interdigittimer"
                               type="xs:integer"
                               use="optional"/>
                 <xs:attribute name="criticaldigittimer"
                               type="xs:integer"
                               use="optional"/>
                 <xs:attribute name="extradigittimer"
                               type="xs:integer"
                               use="optional"/>
                 <xs:attribute name="longtimer"
                               type="xs:integer"
                               use="optional"/>
                 <xs:attribute name="longrepeat"
                               type="xs:boolean"
                               use="optional"/>
               </xs:complexType>
              </xs:element>
```


ATTORNEY DOCKET NO. SNSH-016XX
WEINGARTEN, SCHURGIN,
GAGNEBIN & LEBOVICI LLP
TEL. (617) 542-2290
FAX. (617) 451-0313

```
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="response">
          <xs:complexType>
            <xs:attribute name="code" type="xs:string"
                          use="required"/>
            <xs:attribute name="text" type="xs:string"
                          use="required"/>
            <xs:attribute name="suppressed" type="xs:boolean"
                          use="optional"/>
            <xs:attribute name="forced_flush" type="xs:boolean"
                          use="optional"/>
            <xs:attribute name="digits" type="xs:string"
                          use="optional"/>
            <xs:attribute name="tag" type="xs:string" use="optional"/>
          </xs:complexType>
        </xs:element>
      </xs:choice>
      <xs:attribute name="version" type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Example 10 - Representative KPML Schema

Although in the foregoing description, the particular form of user input constitutes key presses from a telephone keypad, it will be appreciated that the disclosed system and method are also usable with other forms of user input and other devices. For example, the user input may constitute certain patterns of speech that can be recognized by speech recognition hardware/software and then made the subject of subscriptions and notifications as

ATTORNEY DOCKET NO. SNSH-016XX
WEINGARTEN, SCHURGIN,
GAGNEBIN & LEBOVICI LLP
TEL. (617) 542-2290
FAX. (617) 451-0313

described above. The disclosed system and method may also be used with devices such as personal computers (PCs), personal digital assistants (PDAs), or other devices capable of accepting user input in the form of touches or strokes on a screen using a finger or a stylus. Additionally, the "user" generating the input need not be a human user, as in the case of a computer or other device generating key press tones or other user input in a manner mimicking the behavior of a human user.

It will also be apparent to those skilled in the art that other modifications to and variations of the disclosed methods and apparatus are possible without departing from the inventive concepts disclosed herein, and therefore the invention should not be viewed as limited except to the full scope and spirit of the appended claims.

ATTORNEY DOCKET NO. SNSH-016XX
WEINGARTEN, SCHURGIN,
GAGNEBIN & LEBOVICI LLP
TEL. (617) 542-2290
FAX. (617) 451-0313